

FLINT: Fast Lightweight Inference for Traversability

William Bonilla^{1,2,3}, Maxime Boisvert², David-Alexandre Poissant³, David Meger¹, Louis Petit³

Abstract—Navigation in off-road conditions is challenging due to the lack of structure. There is no fixed vocabulary for what is traversable. The traversability depends on both the environment and the embodiment’s dynamics. Neither of these two variables can be hand-labeled at scale. Thus, traversability has to be learned by the embodiment’s own experience. Modern platforms tend to use multiple sensors to estimate traversability and navigate: RGBD cameras, lidar, radar, IMU, with computationally intensive platforms to run inference on neural networks. Against this trend, we propose FLINT, a lightweight traversability estimator: a 21.6M-parameter backbone, 38× smaller than a comparable foundation-model backbone, that scores higher on held-out terrain probes and runs at 14.7 FPS on CPU alone using a RGB camera has the only sensor. Despite that gap in scale, FLINT produces a cheaper, more accurate costmap than a deployed foundation-model system (WildOS) on 23 of 24 replayed field logs. We compare different self-supervised learning signals and deploy the resulting models on a real platform in closed-loop field trials: the best self-supervised head reaches 99% autonomy over the route, outperforming a human-label-trained baseline deployed live on the same course. Our results show that heavy sensing and computing are not necessary for traversability estimation.

Index Terms—Field Robots, Autonomous Vehicle Navigation, Visual-Based Navigation, Learning and Adaptive Systems

I. INTRODUCTION

Traversability is defined as a robot’s ability to cross a terrain region in an admissible state, a concept that depends on the terrain’s geometrical and physical characteristics as well as the specific robot’s capabilities [1]. Field deployments in disaster response, agriculture, and planetary exploration all hinge on getting that judgment right, often with no human able to intervene in time [2]. Off-road traversability estimation is difficult because of how ambiguous the definition can be [3]. It depends on both the terrain and the platform crossing it [4], [5]. To reduce this ambiguity, deployed embodiments are equipped with heavy computing and sensing. For example, V-Strong uses only a single camera, but its full pipeline still requires an NVIDIA A100 GPU [6]. WildOS uses a more modest Jetson Orin, but relies on an extensive sensor suite (3 RGBD cameras + a lidar) [7]. This better sensing alone cannot decide what is traversable and what is not. That capacity requires training labels. The algorithms used require large-scale data to train, and hand-annotated labels at this scale are expensive and

slow. The alternative is self-supervised: learn traversability from what the robot has already driven over [4], [8], [9]. Self-supervised learning is not trivial. Wellhausen et al. [8] label from footholds, BADGR [9] from future collision, and WVN [4] from online visual similarity. Each is a different definition of the same label: a foothold marks a single point the robot’s foot safely touched, a collision label marks a stretch the robot drove without crashing, and a visual-similarity label marks anything that looks like ground already crossed. None of these definitions is obviously the right one, and the choice shapes what the model actually learns. The generated labels do not just describe the terrain, it inherits the capability of the platform that collected them. A robot that can crush a brush will label the brush as traversable, and that label transfers unchanged to a smaller robot that cannot.

This paper makes two contributions:

- **A real-time, CPU-only, single-camera traversability estimator.** A frozen backbone and a lightweight trained head produce dense traversability costmaps without a GPU, lidar, or depth sensor, at a smaller sensing and compute footprint than comparable field systems.
- **A controlled comparison of self-supervision methods.** We compare 3 ways of turning a driven corridor into training labels, on identical data and backbone, against a human-labeled baseline, and evaluate them zero-shot, in closed-loop replay, and live in the field.

We validate both contributions with real robot experiments on a Clearpath Husky A200, using only a single RGB camera, in mixed forest and gravel terrain. In a controlled outdoor deployment, holding route, day, and planner fixed and varying only the traversability head, our best self-supervised heads require fewer interventions than a human-labeled baseline trained specifically for this task. This advantage is not an artifact of one label source: propagating the driven corridor to the rest of the coherent traversable surface, rather than labeling only the corridor itself, is what separates the strongest self-supervised heads from the baseline, and the same ranking reproduces on an independent, offline zero-shot benchmark — pointing to the propagation mechanism itself, rather than any particular label generator, as the source of the gain. Finally, replayed offline against Husky field logs, our heads produce a cheaper driven-path costmap than a deployed foundation-model system, at a fraction of its latency and memory, all from a pipeline that runs on CPU alone showing the result holds even against much larger backbones, not just a human baseline.

¹W. Bonilla and D. Meger are with the School of Computer Science, McGill University, Montreal, QC, Canada. william.bonillavillatoro@mail.mcgill.ca

²W. Bonilla and M. Boisvert are with Centre de Technologies Avancées (CTA), Sherbrooke, QC, Canada.

³W. Bonilla, D.-A. Poissant, and L. Petit are with Université de Sherbrooke, Sherbrooke, QC, Canada.

II. RELATED WORK

Numerous strategies have emerged for teaching robots what terrain they can cross, many built on a different notion of self-supervision, sensor suite, and backbone. We group this body of work around two questions: what signal is used to generate labels from a robot’s own experience, and how much sensing and compute that signal demands to run in the field. Finally, we cover the pre-trained backbones and datasets that anchor these methods.

A. Self-supervised labels for traversability

Self-supervision avoids hand-labeling by treating the terrain a platform has already crossed as a positive traversability label, and numerous approaches have been proposed for off-road autonomy [3]–[6], [8]–[14]. These efforts employ a wide range of modalities to generate that signal. Wellhausen et al. [8] derive it from contact-reprojected footholds a legged robot’s own feet already crossed safely; BADGR [9] instead supervises from the opposite polarity, learning to predict future collision and bumpiness events from IMU and lidar so a policy can steer away from the terrain that produced them, rather than toward terrain already proven safe. WayFAST [10], WayFASTER [11], and Jeon et al. [3] instead track a traction or footprint estimate over RGB-D. Castro et al. [12] and Zürn et al. [13] skip vision entirely, using an IMU-conditioned cost and acoustic terrain sound; Fortin et al. [14] fuse vibration and energy consumption from a UAV-aligned proprioceptive trace. Two are methodologically closest to the strategies we compare later: Wild Visual Navigation (WVN) [4], [5] propagates driven labels to visually similar regions from a short teleoperated demonstration, and V-STRONG [6] derives positive and negative samples from SAM-segmented instances and the driven trajectory. Even the same underlying signal a camera watching where the robot drove yields different information depending on how it is processed: WVN treats it as a similarity cue between regions, V-STRONG as a set of discrete positive and negative instances. Despite this diversity, none have been compared directly on identical data, backbone, and protocol.

B. Computing and sensing

Off-road systems increasingly trade sensing and compute for robustness. V-STRONG [6] and Wild Visual Navigation [4], [5] both need only a single camera, but neither runs without a GPU: V-STRONG’s full pipeline reaches 6 Hz on a single NVIDIA A100, and WVN likewise requires a GPU to run its DINO backbone in real time. WayFAST [10] and WayFASTER [11] target the same real-time, single-robot setting we do, running on an embedded Jetson GPU rather than a workstation-class accelerator, but still depend on a depth channel: both track their traction/footprint estimate over RGB-D, whereas our system runs on a single RGB stream and no GPU at all. WildOS [7] adds heavy sensing on top of heavy compute: an Ouster OS0-128 lidar, a VectorNav VN-100 IMU, and three Intel RealSense D455 RGB-D cameras feed a foundation-model backbone on a Jetson AGX Orin, yet its own reported inference rate is only 1.4 Hz

on that hardware. Boxi [15] goes further still: two lidars, ten RGB cameras, an RGB-D camera, seven IMUs, and a dual-antenna RTK-GNSS receiver twenty-one sensors on one payload.

C. Datasets and backbones

Frozen vision foundation models increasingly serve as backbones for downstream perception without any task-specific fine-tuning. DINOv3 [16] in particular produces dense, patch-level features that transfer across domains without further training, making it attractive as a shared encoder for comparing different downstream strategies on equal footing. Segmentation tools built on this kind of backbone offer complementary views of the same scene: SAM3 [17] provides class-agnostic instance masks with strong object-level boundaries, while STEGO [18] instead produces unsupervised semantic clusters, grouping pixels by learned visual similarity rather than discrete object instances. Together, these tools make it possible to generate structured spatial priors instances or semantic groupings without any manual annotation or model fine-tuning. On the data side, TartanDrive 2.0 [19] offers large-scale, real-world ATV driving data well suited to self-supervised label generation, while RELIS-3D [20] offers densely human-annotated UGV terrain imagery, making it a natural benchmark for evaluating generalization across label sources.

III. METHOD

In this section, we describe how the same driven corridor is turned into training labels under 3 propagation strategies, and how each is used to supervise a lightweight head on a shared frozen backbone. We first detail trajectory reprojection, followed by SAM3 instance propagation and STEGO cluster propagation, each defining positive and negative regions differently from the same underlying signal. We then describe how these labels are used to train the traversability head and how its output is converted into a costmap for deployment.

A. Self-Supervised Label Generation

In this work, we use three propagation strategies: trajectory reprojection (*traj*), SAM3 instance propagation (*sam3*), and STEGO cluster propagation (*stego*), illustrated in Fig. 1 on a single driven frame. Each of the three propagation strategies turns the same driven corridor into a training label for its own version of the MLP head on the frozen DINOv3 [16] backbone. Labels are generated from TartanDrive 2.0 [19], using each run’s own super-odometry and camera calibration to reproject the vehicle’s path. **Trajectory reprojection**, in the spirit of Wild Visual Navigation’s footprint-based supervision [4], [5], projects the vehicle’s future 2.0 s wheel corridor into the image as the positive region; negatives are sampled outside a dilated margin around that corridor, excluding sky and an automatically fitted vehicle-hood mask, and are binned directly onto the 32×32 patch grid. **SAM3 instance propagation**, following V-STRONG’s use of segmentation to extend a sparse trajectory signal [6], uses

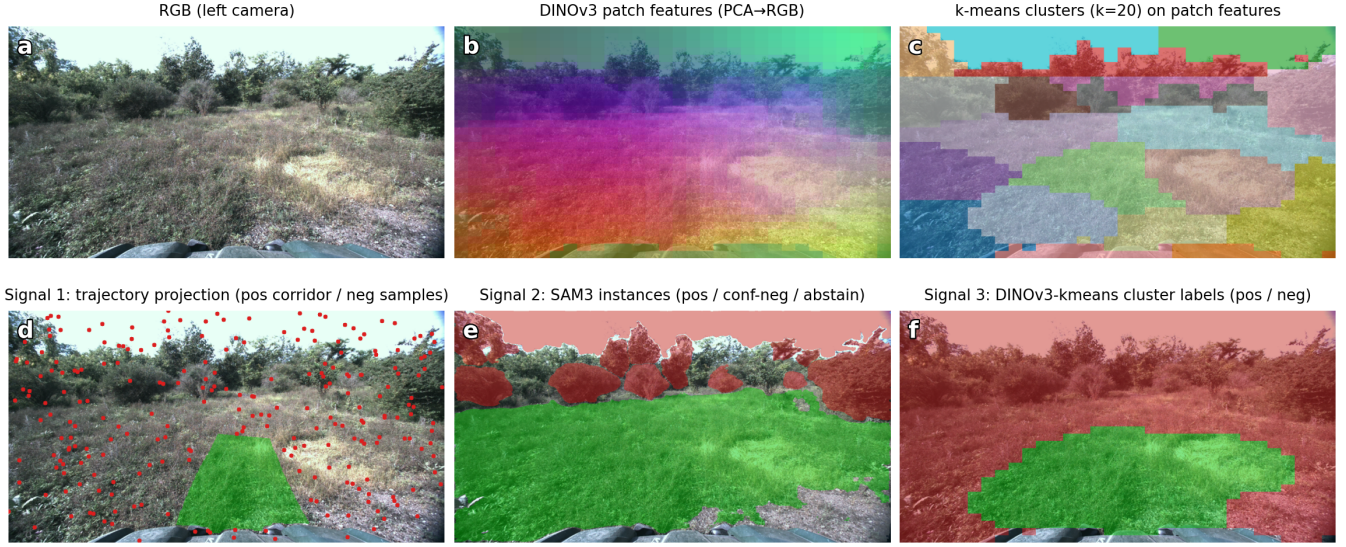


Fig. 1. Label generation on one TartanDrive 2.0 frame (open grass field, `figure_8_morning` run, frame 003058). (a) RGB input; (b) DINOv3 patch features (PCA→RGB); (c) k-means clusters ($k=20$) on those features. The three self-supervision signals: (d) *traj* — reprojected 2 s wheel corridor (green) with sampled negatives (red points) outside a dilated margin; (e) *sam3* — SAM3 instances classified by corridor overlap (green positive, red confident-negative sky/tree, unshaded abstain), which propagates the narrow corridor to the entire drivable surface; (f) *stego* — clusters labeled by corridor overlap on the patch grid. The same drive supervises each head very differently: the corridor band alone (d), the full coherent surface (e), or feature-space clusters (f).

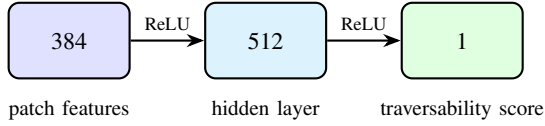


Fig. 2. MLP head architecture: a 384-d frozen DINOv3 patch feature is mapped through a 512-unit hidden layer to a scalar traversability score.

SAM3’s [17] text-prompted instance segmentation over an 11-term terrain vocabulary (dirt trail, rocky ground, smooth dirt road, grass, tall grass, bush, tree, mud, puddle, rock, sky); any instance whose area overlaps the driven corridor by at least 5% is labeled positive in full, propagating the sparse trajectory signal to the entire coherent surface it lies on. Only *sky* is confident negative, labeled negative unconditionally regardless of overlap; every other term that falls short of the overlap threshold abstains rather than being labeled negative, since not having been driven on during one pass is not evidence that, say, a patch of grass is non-traversable in general.

B. Head Training

- 1) **Data and split.** The training data is split by run rather than by frame: adjacent frames from the same drive are highly correlated, so a frame-level split would let a head see near-duplicates of its own validation data during training and report inflated performance. Every head instead trains on the same 49 TartanDrive 2.0 runs (104,083 valid frames of 119,020 raw, stationary segments excluded), split 85/15 *by run* with a fixed seed, so no run leaks between train and validation.

- 2) **Optimization.** Each head is the same MLP head (Fig. 2) with 197,633 trainable parameters, 0.9% of the frozen backbone trained with BCEWithLogitsLoss and AdamW (learning rate 10^{-4} , weight decay 0.01, cosine-annealed) for 20 epochs on a single 12 GB desktop GPU (RTX 5070).
- 3) **Per-head cost.** *sam3* and *stego* train MLP-only on pre-extracted per-frame features (batch 256, minutes per run); *traj* recomputes the frozen backbone at every step, since its labels are purely geometric and not cached.

C. System Overview

FLINT itself comprises the frozen backbone, the active propagation strategy’s MLP head, and the IPM projection step; Nav2 and the underlying hardware are not part of our contribution, and are included in Fig. 3 only to show the complete deployed system.

- **Hardware.** The robot is equipped with a MultiSense S27 stereo camera, already mounted on the Husky prior to this work rather than selected for it, and standard wheel encoders for odometry. We use only the camera’s single rectified RGB stream and never its depth output.
- **Costmap.** Each frame is processed by a frozen DINOv3 ViT-S/16 backbone into a 32×32 grid of patch-level features, from which the active propagation strategy’s head predicts a per-patch traversability probability. These probabilities are projected onto the ground plane via inverse perspective mapping and handed to Nav2 [21], [22] as an OccupancyGrid; Nav2 owns costmap construction, planning, and low-level control from that point on.

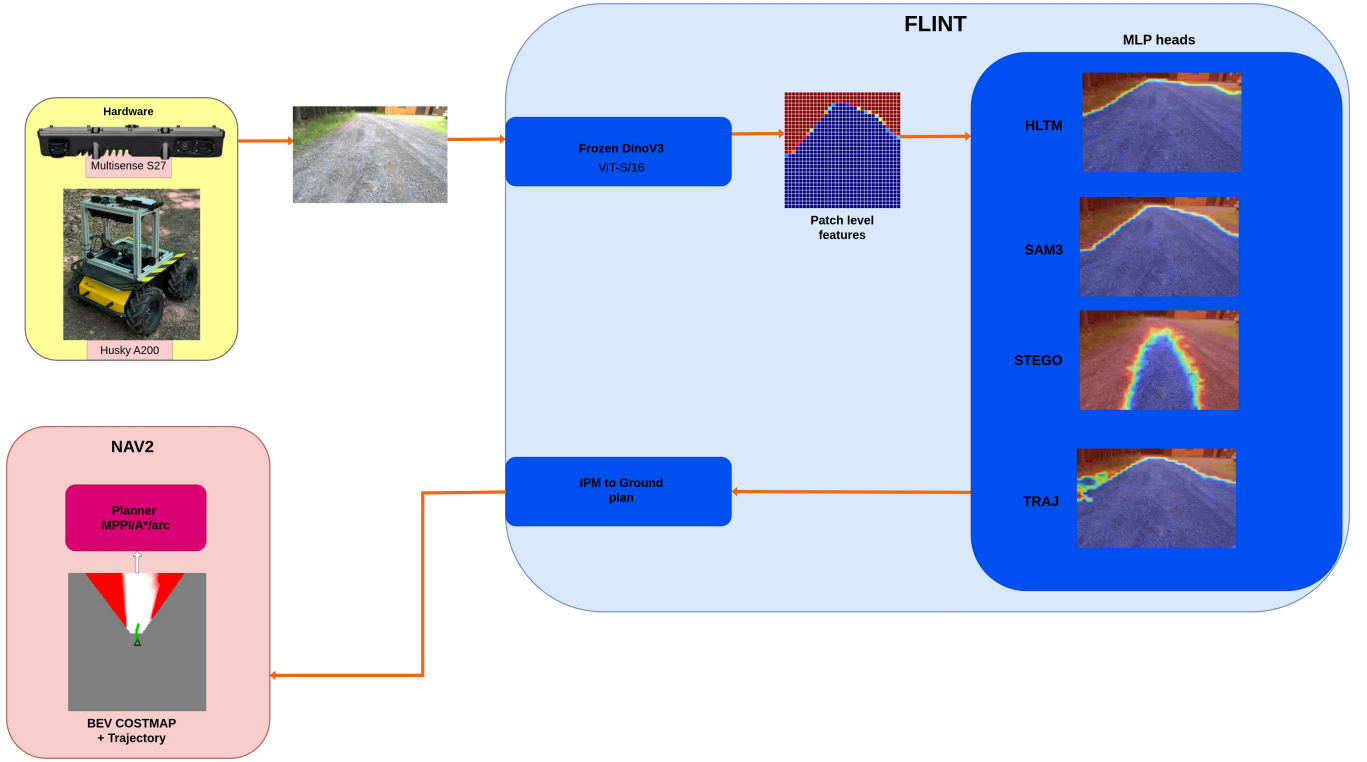


Fig. 3. **System Overview.** A single RGB frame captured by the Husky A200’s Multisense S27 stereo camera is processed by a frozen DINOv3 (ViT-S/16) backbone into patch-level features. Four lightweight MLP heads (HLTM, SAM3, STEGO, TRAJ) independently propagate these features into a dense traversability probability map, shown here for each head on real field data (blue = high traversability probability, orange = low). The active head’s probabilities are passed through an inverse perspective mapping (IPM) module to produce a ground-plane occupancy grid, which Nav2 consumes as a BEV costmap to generate a planned trajectory (white corridor) via its MPPI/A*/arc planner. Colored regions denote organizational groupings only (hardware, FLINT, Nav2), not a claim of novelty.

- **Planner.** The costmap is planner-agnostic; any Nav2-compatible planner can consume it without retraining or modifying FLINT.
- **Onboard Computation.** In deployment, exactly one head is active at a time. The entire forward pass, backbone and that one head, runs on CPU alone at 68 ms/frame (14.7 FPS). Because the backbone is frozen and shared across heads, its 68 ms cost is paid once regardless of how many traversability heads are evaluated on the same frame; each additional lightweight head adds negligible overhead on top, a property this paper exploits only for the offline side-by-side comparisons of Sec. IV-A IV-C, not in the live field system, which always runs a single head.

D. Human-Labeled Baseline

We include HLTM to establish how far self-supervised labels can close the gap to human annotation, the offline accuracy ceiling this paper’s self-supervised heads are measured against; Sec. IV-B analyzes this comparison directly, both in live closed-loop deployment and offline zero-shot evaluation. HLTM shares the same frozen DINOv3 backbone and MLP head as the three self-supervised heads above, but its label comes from human annotation rather than driven-corridor propagation: dense per-pixel traversability masks pooled across three human-labeled datasets (Yamaha [23], 1,076

images; CAVS [24], 677 images; Great Outdoors [25], 1,199 images), collapsed to a binary traversable/non-traversable label under the same convention used to build the RELLIS-3D evaluation ground truth. RELLIS-3D itself is held out entirely from this training pool, so HLTM’s RELLIS-3D evaluation is genuinely zero-shot, exactly like the other three heads.

IV. EXPERIMENTS

This section evaluates FLINT along three axes: how it compares against heavier foundation-model baselines, how its self-supervised heads compare against a human-labeled baseline in live deployment, and why the winning self-supervised heads win. Sec. IV-A first establishes that FLINT is competitive with a much larger foundation-model system, before turning to comparisons among our own heads. Sec. IV-B then presents this paper’s headline field result: two self-supervised heads outperform a human-labeled baseline in live closed-loop deployment. Sec. IV-C explains that result mechanistically, bringing the weakest self-supervised head back into the comparison to show what separates it from the two winners. Sec. IV-D closes with the compute cost behind all of the above, confirming the entire pipeline sustained real-time inference on CPU alone throughout every field deployment.

TABLE I
OFFLINE COMPARISON AGAINST FOUNDATION-MODEL BASELINES

System	Mean pct.	<WildOS	ms/frame	GPU MB
stego	18.1	23/24	70.6	391
traj	23.8	23/24	70.6	391
sam3	26.5	22/24	70.6	391
WildOS [7]	41.0	—	405.3	1045
WVN (shadow) [4], [5]	54.2	4/24	44.0	382

A. Offline Comparison Against Foundation-Model Baselines

Before comparing self-supervision strategies against each other, we first ask whether FLINT is competitive with state-of-the-art traversability estimators at all. Live, closed-loop deployment of two natural baselines was not feasible on our platform: WildOS [7]’s full stack assumes a sensor suite (3 RGB-D cameras + a lidar) our Husky A200 does not carry, and Wild Visual Navigation (WVN) [4], [5]’s learning node is implemented in ROS1, with no straightforward path onto our ROS2/Nav2 stack. Instead, we ran WildOS’s own inference node for traversability estimation (ExploRFM) directly against RGB alone, excluding its lidar and elevation-mapping stages, to isolate the vision module both systems actually share, and built a native, ROS-independent reimplementaion of WVN’s core online-learning algorithm, run in shadow mode (no actuation) but still taking real gradient steps from the log’s own velocity-tracking error. Both were replayed against 24 Husky field logs spanning five collection days, alongside our own three heads served from one shared-backbone process, and every resulting costmap was scored with the same threshold-free driven-cost percentile used throughout this paper: the actually-driven path is a fact independent of any head’s own calibration, so the metric holds even though WildOS, WVN, and our three heads disagree about what to do with unfamiliar terrain.

Table I gives the result: all three of our heads produce a cheaper driven-path costmap than WildOS in the large majority of logs (traj/stego 23/24, sam3 22/24), at $5.7\times$ lower latency on the deployed Jetson Xavier and roughly a third of its GPU memory sharing one 70.6 ms backbone pass across all three heads where WildOS pays 405.3 ms for one. WVN-shadow is competitive on latency and memory but beats WildOS on only 4 of 24 logs, sitting close to the random-path level on several of them, a caveat worth noting before treating it as a strong baseline elsewhere. This establishes that FLINT is not merely cheap: on this deployed-costmap metric it is also competitive with, and typically better than, a foundation-model system built for the same task, despite FLINT’s backbone being $38\times$ smaller and needing a lighter sensor suite. With that established, the rest of this section turns to the comparison this paper is actually about: which way of turning a driven corridor into a self-supervised label works best.

B. Self-Supervised Heads vs. Human-Labeled Baseline

Before presenting this section’s result, one definition needs to be fixed: what counts as an intervention. We define it as

any interval in which the operator disengages autonomous mode via the planner’s deadman switch, regardless of cause. We further distinguish two categories by manually reviewing the costmap and driven context at each episode, a **minor** intervention being a brief correction issued while the active head’s traversability estimate was locally correct, needed only because the planner could not resolve a sharp turn or trail junction from within the costmap’s current field of view, and a **major** intervention being a takeover triggered by an actual traversability misclassification, where the active head’s costmap marked an unsafe region as traversable (or the reverse), the planner routed the robot toward the resulting hazard, and the operator had to redirect it away, a distinction made manually from the recorded costmap and video rather than an automatically logged signal, and one that reported intervention counts and intervention-free distances below aggregate across unless noted otherwise

To isolate the effect of the traversability head from all other system variables, we deployed all four candidate heads (traj, sam3, stego, and a human-label-trained model, HLTM) on the same route, on the same day, using the identical A* planner and costmap interface; only the head producing the traversability estimate differed between runs. Across the four runs, the robot covered a combined 4.47 km, accumulating 119.1 min of continuous autonomous-mode CPU-only inference with no dropped frames. Fig. 4 overlays sam3, stego, and HLTM’s driven GPS trajectories on a shared basemap, with clustered operator-intervention episodes marked; traj’s trajectory is omitted from this figure for legibility (frequent interventions fragmented its track into many short, overlapping segments) but its full results are in Table III along with the other three.

Table III gives the result of this paper’s field comparison: the two self-supervised heads, sam3 and stego, both outperformed HLTM. sam3 required the fewest interventions relative to distance and reached the highest autonomy rate (99.0%), while stego produced the single longest intervention-free stretch of any head, 809.9 m (21.9 min) continuous. Notably, both self-supervised heads outperformed HLTM in live closed-loop deployment (96.3% autonomy, 7 interventions), despite HLTM being the strongest head, zero-shot, on the offline RELLIS-3D benchmark (Table II). We evaluate on RELLIS-3D [20] both because V-STRONG [6] reports its own results on the same dataset, letting us situate our numbers against a prior system on common ground, and because its dense, human-annotated ground truth makes it a default standard benchmark for off-road traversability. Self-supervision did not just approach human-labeled quality; it exceeded it once deployed on the platform that generated its own labels. This ranking holds up under a second, independent measure. We define driven-cost percentile as

$$p = \frac{|\{i : c_i \leq c_{\text{driven}}\}|}{N} \times 100, \quad (1)$$

where c_{driven} is the cost of the head’s actually-driven path and $\{c_1, \dots, c_N\}$ ($N=100$) are the costs of random, kinematically-feasible alternative paths sampled from the



Fig. 4. GPS trajectories for stego, sam3 and HLTM heads on the identical route and planner, overlaid on a shared basemap. Clustered operator-intervention episodes are marked per head. Added below are example interventions: successful navigation, a minor intervention near the trail edge, and a major intervention into high grass.

same start over the same horizon; a value of 0 means the driven path was cheaper than every alternative, 50 is indistinguishable from a random path, and 100 means it was the most expensive path tried. On the same deployment, this places sam3 (13.9%) ahead of scout_human/HLTMs (15.0%), itself ahead of stego (16.6%); this is the same best-to-worst ordering, among these three heads, as the offline zero-shot RELLIS-3D evaluation. Two independently-designed protocols, one offline and one live closed-loop, agreeing on the same ranking is corroborating evidence rather than coincidence.

C. Head-to-Head Among Self-Supervised Signals

traj was left out of Fig. 4 so that result would land cleanly. traj was the weakest head by every measure: 87.5% autonomy, 8 interventions live, and 22.1% on the driven-cost-percentile analysis above, worst on both protocols, and worst in the offline zero-shot RELLIS-3D ranking. The reason is what traj does *not* do: its label only ever covers the driven corridor itself, a fixed-width band the vehicle happened to occupy, and never propagates to the rest of the visibly coherent traversable surface around it. sam3 and stego are best understood as a more structured version

TABLE II
ZERO-SHOT RELLIS-3D EVALUATION (IoU AT FIXED THRESHOLD $\tau = 0.5$).

model	superv.	AUROC	AUPRC	IoU
traj	self, geom.	0.639	0.447	0.366
stego	self, clust.	0.948	0.911	0.418
sam3	self, SAM3	0.956	0.918	0.749
HLTMs	human	0.985	0.963	0.824

TABLE III
SELF-SUPERVISED HEADS VS. THE HUMAN-LABELED BASELINE, IDENTICAL ROUTE AND PLANNER.

Head	Dist. (m)	Auton. %	Longest (m)	Longest (s)	Interv.
sam3	1125.7	99.0	398.3	639.8	3
stego	1059.7	97.7	809.9	1313.4	3
HLTMs	1114.5	96.3	279.8	454.8	7
traj	1165.2	87.5	197.0	315.8	8

of that same raw signal: sam3 propagates it through SAM3's

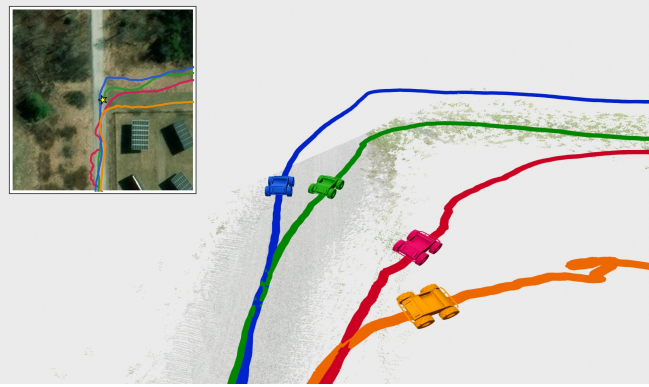


Fig. 5. 3D reconstruction of the route’s sharpest turn, colored by head, with satellite inset showing the driven path

instance segmentation masks, and `stego` through STEGO’s superpixel-like feature clusters, and it is that additional structure, not a better label source, that lets them close the gap to HLTM and then exceed it. Put differently, Sec. IV-B’s headline result is not a property of self-supervision generically; it is a property specifically of self-supervision that propagates beyond the driven band, and `traj` is the control condition that shows what is lost without it. This is also evidence for a broader point: traversability is a property of a region, not just the exact footprint a vehicle happened to occupy, and `traj`’s failure is the direct demonstration of that. That `sam3` and `stego` can recover this broader region from nothing but a frozen backbone and a driven corridor is itself consistent with a known property of DINO-style self-supervised features: they carry explicit segmentation structure that does not emerge as clearly from supervised training or direct segmentation objectives [26], and our result is one more data point for that claim in a domain, off-road terrain, far from where it was first observed.

Fig. 5 makes a related point concretely rather than statistically. All four heads find this turn traversable, `stego`, `sam3`, and `traj` cross it without intervention, and HLTM needs only one small correction, yet each settles onto a physically different line through the identical turn, from `stego`’s tight inside cut to HLTM’s wide outside arc. There is no single correct line here: all four solutions work. What differs is not whether the terrain is traversable but how each head’s training shapes the specific path chosen through it, evidence that the training signal leaves a real, physical fingerprint on behavior even when every resulting trajectory succeeds.

D. CPU-Only Compute Benchmark

Unlike a bench-only latency figure (68.2ms/frame, 14.7FPS), this throughput was sustained live: FLINT ran CPU-only inference in the robot’s real-time control loop for the entirety of all four field deployments, 119.1min of continuous autonomous-mode inference over 4.47km combined, with no dropped frames or fallback to a lower rate. The same inference code, run on both devices of one desktop machine, confirms this is not a lower bound in principle: 68.2ms/frame (14.7FPS) on CPU alone (16

threads) versus 12.3ms/frame (81.5FPS) on its GPU (RTX 5070), comfortably above the field system’s real-time floor either way. GPU-free deployment is a live option on the real-time control computer, not just something that happens to work on desktop hardware.

We separately benchmarked WVN’s [4], [5] own feature-extraction pipeline on the `sam3` field bags, forced onto CPU rather than its default GPU path: it processed frames at a mean 832.7ms/frame (1.20Hz), **12.2× slower than FLINT’s 68.2ms/frame** on the same CPU hardware, a lower bound on its true cost since its online-learning step was not confirmed active during measurement.

V. CONCLUSION

We presented FLINT, a traversability estimator built on a frozen, 21.6M-parameter backbone 38× smaller than a comparable foundation-model backbone that runs at 14.7FPS on CPU alone, requiring no GPU for inference, and roughly 12× faster than a comparable prior system’s feature-extraction pipeline when both are run CPU-only. The system sustained real-time CPU-only inference live, in the robot’s control loop, for 119.1min of continuous autonomous driving across 4.47km. We compared three self-supervised label sources against a human-labeled baseline in a controlled head-to-head field deployment, holding the route and planner fixed and varying only the traversability head. The best self-supervised head reached 99.0% autonomy and outperformed the human-labeled baseline in live deployment, despite the latter beating every self-supervised head zero-shot on RELLIS-3D. An independent driven-cost-percentile analysis of the same deployment agrees with our offline zero-shot ranking on RELLIS-3D, corroborating the result across two independently-designed evaluations. A qualitative comparison at a single shared, zero-intervention waypoint further shows that heads can agree on safe outcomes while disagreeing on the underlying traversability estimate; multiple valid representations can each support safe autonomy without converging to the same judgment. These results indicate that reliable traversability estimation does not require heavy sensing or compute, and that self-supervised labels collected by the deploying platform itself can match or exceed hand-labeled supervision once evaluated in closed loop. Limitations of this study include a single test route and single-day conditions. Future work includes testing across a wider range of terrain and conditions, and closing the loop with online adaptation of the traversability head during deployment.

ACKNOWLEDGMENT

Portions of this manuscript’s prose (revision throughout all sections), figure-generation, and table formatting were produced with the assistance of Claude (Anthropic), under continuous author direction and review. All experimental design, data collection, code, results, and technical claims were carried out and verified by the authors.

REFERENCES

- [1] A. Gholami and A. Ramirez-Serrano, "Terrain traversability via sensed data for robots operating inside heterogeneous, highly unstructured spaces," *Sensors*, vol. 25, no. 2, p. 439, 2025. [Online]. Available: <https://doi.org/10.3390/s25020439>
- [2] S. Beycimen, D. Ignatyev, and A. Zolotas, "A comprehensive survey of unmanned ground vehicle terrain traversability for unstructured environments and sensor technology insights," *Engineering Science and Technology, an International Journal*, vol. 47, p. 101457, 2023.
- [3] Y. Jeon, E. I. Son, and S.-W. Seo, "Follow the footprints: Self-supervised traversability estimation for off-road vehicle navigation based on geometric and visual cues," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 1774–1780.
- [4] J. Frey, M. Mattamala, N. Chebrolu, C. Cadena, M. Fallon, and M. Hutter, "Fast traversability estimation for wild visual navigation," in *Robotics: Science and Systems (RSS)*, 2023.
- [5] M. Mattamala, J. Frey, P. Libera, N. Chebrolu, G. Martius, C. Cadena, M. Hutter, and M. Fallon, "Wild visual navigation: Fast traversability learning via pre-trained models and online self-supervision," *Autonomous Robots*, 2025.
- [6] S. Jung, J. Lee, X. Meng, B. Boots, and A. Lambert, "V-strong: Visual self-supervised traversability learning for off-road navigation," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 1766–1773.
- [7] H. Shah, E. Tevere, D. Atha, M. Kaufmann, S. Khattak, M. Patel, M. Hutter, J. Frey, and P. Spieler, "WildOS: Open-vocabulary object search in the wild," *arXiv preprint arXiv:2602.19308*, 2026.
- [8] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where should i walk? Predicting terrain properties from images via self-supervised learning," *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, pp. 1509–1516, 2019.
- [9] G. Kahn, P. Abbeel, and S. Levine, "BADGR: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters (RA-L)*, 2021, also presented at ICRA 2021.
- [10] M. V. Gasparino, A. N. Sivakumar, Y. Liu, A. E. B. Velasquez, V. A. H. Higuti, J. Rogers, H. Tran, and G. Chowdhary, "WayFAST: Navigation with predictive traversability in the field," *IEEE Robotics and Automation Letters (RA-L)*, 2022.
- [11] M. V. Gasparino, A. N. Sivakumar, and G. Chowdhary, "WayFASTER: A self-supervised traversability prediction for increased navigation awareness," in *IEEE ICRA*, 2024.
- [12] M. G. Castro, S. Triest, W. Wang, J. M. Gregory, F. Sanchez, J. G. Rogers, and S. Scherer, "How does it feel? Self-supervised costmap learning for off-road vehicle traversability," in *IEEE ICRA*, 2023, pp. 931–938.
- [13] J. Zörn, W. Burgard, and A. Valada, "Self-supervised visual terrain classification from unsupervised acoustic feature learning," *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 466–481, 2020.
- [14] J.-M. Fortin, O. Gamache, W. Fecteau, E. Daum, W. Larrivée-Hardy, F. Pomerleau, and P. Giguère, "UAV-assisted self-supervised terrain awareness for off-road navigation," in *IEEE ICRA*, 2025.
- [15] J. Frey, T. Tuna, L. F. T. Fu, C. Weibel, K. Patterson, B. Krummenacher, M. Müller, J. Nubert, M. Fallon, C. Cadena, and M. Hutter, "Boxi: Design Decisions in the Context of Algorithmic Performance for Robotics," in *Proceedings of Robotics: Science and Systems*, Los Angeles, United States, July 2025, *Equal contribution (Jonas Frey and Turcan Tuna and Frank Fu).
- [16] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa *et al.*, "DINOv3," *arXiv preprint arXiv:2508.10104*, 2025.
- [17] N. Carion, L. Gustafson, Y.-T. Hu, S. Debnath, R. Hu, D. Suris, C. Ryali, K. V. Alwala, H. Khedr, P. Dollár, N. Ravi, K. Saenko, P. Zhang, C. Feichtenhofer *et al.*, "SAM 3: Segment anything with concepts," *arXiv preprint arXiv:2511.16719*, 2025.
- [18] M. Hamilton, Z. Zhang, B. Hariharan, N. Snavely, and W. T. Freeman, "Unsupervised semantic segmentation by distilling feature correspondences," in *ICLR*, 2022.
- [19] M. Sivaprakasam *et al.*, "TartanDrive 2.0: More modalities and better infrastructure to further self-supervised learning research in off-road driving tasks," *IEEE ICRA*, 2024.
- [20] P. Jiang, P. Osteen, M. Wigness, and S. Saripalli, "RELLIS-3D dataset: Data, benchmarks and analysis," in *IEEE ICRA*, 2021.
- [21] S. Macenski, F. Martin, R. White, and J. G. Clavero, "The Marathon 2: A navigation system," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- [22] S. Macenski, T. Moore, D. V. Lu, A. Merzlyakov, and M. Ferguson, "From the desks of ROS maintainers: A survey of modern & capable mobile robotics algorithms in the robot operating system 2," *Robotics and Autonomous Systems*, 2023.
- [23] D. Maturana, P.-W. Chou, M. Uenoyama, and S. Scherer, "Real-time semantic mapping for autonomous off-road navigation," in *Field and Service Robotics (FSR)*. Springer, 2018, pp. 335–350.
- [24] L. Dabbiru, S. Sharma, K. A. Ennin, C. T. Goodin, C. R. Hudson, M. Doude, D. W. Carruth, and J. E. Ball, "CAVS semantic segmentation dataset for off-road autonomous driving," in *Proc. SPIE 13474, Autonomous Systems: Sensors, Processing, and Security for Ground, Air, Sea, and Space Vehicles and Infrastructure*, 2025.
- [25] P. Jiang, K. Viswanath, A. Nagariya, G. Chustz, M. Wigness, P. Osteen, T. Overbye, C. Ellis, L. Quang, J. Huang, and S. Saripalli, "GO: The great outdoors multimodal dataset," in *IEEE Intelligent Vehicles Symposium (IV)*, 2026.
- [26] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9650–9660.